# MAVEN BUILD LIFE CYCLE

 

## What is Build Lifecycle?

A *Build Lifecycle* is a well defined sequence of phases which define the order in which the goals are to be executed. Here phase represents a stage in life cycle.

As an example, a typical *Maven Build Lifecycle* is consists of following sequence of phases

| Phase | Handles | Description |
|---|---|---|
| prepare-resources | resource copying | Resource copying can be customized in this phase. |
| compile | compilation | Source code compilation is done in this phase. |
| package | packaging | This phase creates the JAR / WAR package as mentioned in packaging in POM.xml. |
| install | installation | This phase installs the package in local / remote maven repository. |

There are always **pre** and **post** phases which can be used to register **goals** which must run prior to or after a particular phase.

When Maven starts building a project, it steps through a defined sequence of phases and executes goals which are registered with each phase. Maven has following three standard lifecycles:

- clean
- default(or build)
- site

A **goal** represents a specific task which contributes to the building and managing of a project. It may be bound to zero or more build phases. A goal not bound to any build phase could be executed outside of the build lifecycle by direct invocation.

The order of execution depends on the order in which the goal(s) and the build phase(s) are invoked. For example, consider the command below. The clean and package arguments are build phases while the *dependency:copy-dependencies* is a goal.

```
mvn clean dependency:copy-dependencies package
```

Here the *clean* phase will be executed first, and then the *dependency:copy-dependencies goal* will be executed, and finally *package* phase will be executed.

## Clean Lifecycle

When we execute *mvn post-clean* command, Maven invokes the clean lifecycle consisting of the following phases.

- pre-clean

- clean
- post-clean

Maven clean goal (clean:clean) is bound to the *clean* phase in the clean lifecycle. Its *clean:clean* goal deletes the output of a build by deleting the build directory. Thus when *mvn clean* command executes, Maven deletes the build directory.

We can customize this behavior by mentioning goals in any of the above phases of clean life cycle.

In the following example, We'll attach maven-antrun-plugin:run goal to the pre-clean, clean, and post-clean phases. This will allow us to echo text messages displaying the phases of the clean lifecycle.

We've created a pom.xml in C:\MVN\project folder.

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
   http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>com.companyname.projectgroup</groupId>
<artifactId>project</artifactId>
<version>1.0</version>
<build>
<plugins>
   <plugin>
   <groupId>org.apache.maven.plugins</groupId>
   <artifactId>maven-antrun-plugin</artifactId>
   <version>1.1</version>
   <executions>
      <execution>
         <id>id.pre-clean</id>
         <phase>pre-clean</phase>
         <goals>
            <goal>run</goal>
         </goals>
         <configuration>
            <tasks>
               <echo>pre-clean phase</echo>
            </tasks>
         </configuration>
      </execution>
      <execution>
         <id>id.clean</id>
         <phase>clean</phase>
         <goals>
          <goal>run</goal>
         </goals>
         <configuration>
            <tasks>
               <echo>clean phase</echo>
            </tasks>
         </configuration>
      </execution>
      <execution>
         <id>id.post-clean</id>
         <phase>post-clean</phase>
         <goals>
            <goal>run</goal>
         </goals>
         <configuration>
            <tasks>
               <echo>post-clean phase</echo>
            </tasks>
         </configuration>
      </execution>
   </executions>
   </plugin>
</plugins>
</build>
```

```
</project>
```

Now open command console, go to the folder containing pom.xml and execute the following **mvn** command.

```
C:\MVN\project>mvn post-clean
```

Maven will start processing and display all the phases of clean life cycle

```
[INFO] Scanning for projects...
[INFO] ------------------------------------------------------------------------
[INFO] Building Unnamed - com.companyname.projectgroup:project:jar:1.0
[INFO]    task-segment: [post-clean]
[INFO] ------------------------------------------------------------------------
[INFO] [antrun:run {execution: id.pre-clean}]
[INFO] Executing tasks
     [echo] pre-clean phase
[INFO] Executed tasks
[INFO] [clean:clean {execution: default-clean}]
[INFO] [antrun:run {execution: id.clean}]
[INFO] Executing tasks
     [echo] clean phase
[INFO] Executed tasks
[INFO] [antrun:run {execution: id.post-clean}]
[INFO] Executing tasks
     [echo] post-clean phase
[INFO] Executed tasks
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESSFUL
[INFO] ------------------------------------------------------------------------
[INFO] Total time: < 1 second
[INFO] Finished at: Sat Jul 07 13:38:59 IST 2012
[INFO] Final Memory: 4M/44M
[INFO] ------------------------------------------------------------------------
```

You can try tuning **mvn clean** command which will display *pre-clean* and *clean*, nothing will be executed for *post-clean* phase.

## Default (or Build) Lifecycle

This is the primary life cycle of Maven and is used to build the application. It has following 23 phases.

| Lifecycle Phase | Description |
|---|---|
| validate | Validates whether project is correct and all necessary information is available to complete the build process. |
| initialize | Initializes build state, for example set properties |
| generate-sources | Generate any source code to be included in compilation phase. |
| process-sources | Process the source code, for example, filter any value. |
| generate-resources | Generate resources to be included in the package. |
| process-resources | Copy and process the resources into the destination directory, ready for packaging phase. |
| compile | Compile the source code of the project. |
| process-classes | Post-process the generated files from compilation, for example to do bytecode enhancement/optimization on Java classes. |

| | |
|---|---|
| generate-test-sources | Generate any test source code to be included in compilation phase. |
| process-test-sources | Process the test source code, for example, filter any values. |
| test-compile | Compile the test source code into the test destination directory. |
| process-test-classes | Process the generated files from test code file compilation. |
| test | Run tests using a suitable unit testing framework(Junit is one). |
| prepare-package | Perform any operations necessary to prepare a package before the actual packaging. |
| package | Take the compiled code and package it in its distributable format, such as a JAR, WAR, or EAR file. |
| pre-integration-test | Perform actions required before integration tests are executed. For example, setting up the required environment. |
| integration-test | Process and deploy the package if necessary into an environment where integration tests can be run. |
| pre-integration-test | Perform actions required after integration tests have been executed. For example, cleaning up the environment. |
| verify | Run any check-ups to verify the package is valid and meets quality criterias. |
| install | Install the package into the local repository, which can be used as a dependency in other projects locally. |
| deploy | Copies the final package to the remote repository for sharing with other developers and projects. |

There are few important concepts related to Maven Lifecycles which are wroth to mention:

- When a phase is called via Maven command, for example *mvn compile*, only phases upto and including that phase will execute.
- Different maven goals will be bound to different phases of Maven lifecycle depending upon the type of packaging (JAR / WAR / EAR).

In the following example, We'll attach maven-antrun-plugin:run goal to few of the phases of Build lifecycle. This will allow us to echo text messages displaying the phases of the lifecycle.

We've updated pom.xml in C:\MVN\project folder.

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>com.companyname.projectgroup</groupId>
<artifactId>project</artifactId>
<version>1.0</version>
<build>
<plugins>
<plugin>
<groupId>org.apache.maven.plugins</groupId>
```

```
<artifactId>maven-antrun-plugin</artifactId>
<version>1.1</version>
<executions>
   <execution>
      <id>id.validate</id>
      <phase>validate</phase>
      <goals>
         <goal>run</goal>
      </goals>
      <configuration>
         <tasks>
            <echo>validate phase</echo>
         </tasks>
      </configuration>
   </execution>
   <execution>
      <id>id.compile</id>
      <phase>compile</phase>
      <goals>
         <goal>run</goal>
      </goals>
      <configuration>
         <tasks>
            <echo>compile phase</echo>
         </tasks>
      </configuration>
   </execution>
   <execution>
      <id>id.test</id>
      <phase>test</phase>
      <goals>
         <goal>run</goal>
      </goals>
      <configuration>
         <tasks>
            <echo>test phase</echo>
         </tasks>
      </configuration>
   </execution>
   <execution>
         <id>id.package</id>
         <phase>package</phase>
         <goals>
            <goal>run</goal>
         </goals>
         <configuration>
         <tasks>
            <echo>package phase</echo>
         </tasks>
      </configuration>
   </execution>
   <execution>
      <id>id.deploy</id>
      <phase>deploy</phase>
      <goals>
         <goal>run</goal>
      </goals>
      <configuration>
      <tasks>
         <echo>deploy phase</echo>
      </tasks>
      </configuration>
   </execution>
</executions>
</plugin>
</plugins>
</build>
</project>
```

Now open command console, go the folder containing pom.xml and execute the following **mvn** command.

```
```

```
C:\MVN\project>mvn compile
```

Maven will start processing and display phases of build life cycle upto compile phase.

```
[INFO] Scanning for projects...
[INFO] ------------------------------------------------------------------------
[INFO] Building Unnamed - com.companyname.projectgroup:project:jar:1.0
[INFO]    task-segment: [compile]
[INFO] ------------------------------------------------------------------------
[INFO] [antrun:run {execution: id.validate}]
[INFO] Executing tasks
     [echo] validate phase
[INFO] Executed tasks
[INFO] [resources:resources {execution: default-resources}]
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources,
i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory C:\MVN\project\src\main\resources
[INFO] [compiler:compile {execution: default-compile}]
[INFO] Nothing to compile - all classes are up to date
[INFO] [antrun:run {execution: id.compile}]
[INFO] Executing tasks
     [echo] compile phase
[INFO] Executed tasks
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESSFUL
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 2 seconds
[INFO] Finished at: Sat Jul 07 20:18:25 IST 2012
[INFO] Final Memory: 7M/64M
[INFO] ------------------------------------------------------------------------
```

## Site Lifecycle

Maven Site plugin is generally used to create fresh documentation to create reports, deploy site etc.

Phases

- pre-site
- site
- post-site
- site-deploy

In the following example, We'll attach *maven-antrun-plugin:run* goal to all the phases of Site lifecycle. This will allow us to echo text messages displaying the phases of the lifecycle.

We've updated pom.xml in C:\MVN\project folder.

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>com.companyname.projectgroup</groupId>
<artifactId>project</artifactId>
<version>1.0</version>
<build>
<plugins>
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-antrun-plugin</artifactId>
<version>1.1</version>
   <executions>
      <execution>
         <id>id.pre-site</id>
```

```
         <phase>pre-site</phase>
         <goals>
            <goal>run</goal>
         </goals>
         <configuration>
            <tasks>
               <echo>pre-site phase</echo>
            </tasks>
         </configuration>
      </execution>
      <execution>
         <id>id.site</id>
         <phase>site</phase>
         <goals>
         <goal>run</goal>
         </goals>
         <configuration>
            <tasks>
               <echo>site phase</echo>
            </tasks>
         </configuration>
      </execution>
      <execution>
         <id>id.post-site</id>
         <phase>post-site</phase>
         <goals>
            <goal>run</goal>
         </goals>
         <configuration>
            <tasks>
               <echo>post-site phase</echo>
            </tasks>
         </configuration>
      </execution>
      <execution>
         <id>id.site-deploy</id>
         <phase>site-deploy</phase>
         <goals>
            <goal>run</goal>
         </goals>
         <configuration>
            <tasks>
               <echo>site-deploy phase</echo>
            </tasks>
         </configuration>
      </execution>
   </executions>
</plugin>
</plugins>
</build>
</project>
```

Now open command console, go the folder containing pom.xml and execute the following **mvn** command.

```
C:\MVN\project>mvn site
```

Maven will start processing and display phases of site life cycle upto site phase.

```
[INFO] Scanning for projects...
[INFO] ------------------------------------------------------------------------
[INFO] Building Unnamed - com.companyname.projectgroup:project:jar:1.0
[INFO]    task-segment: [site]
[INFO] ------------------------------------------------------------------------
[INFO] [antrun:run {execution: id.pre-site}]
[INFO] Executing tasks
     [echo] pre-site phase
[INFO] Executed tasks
[INFO] [site:site {execution: default-site}]
[INFO] Generating "About" report.
[INFO] Generating "Issue Tracking" report.
```

```
[INFO] Generating "Project Team" report.
[INFO] Generating "Dependencies" report.
[INFO] Generating "Project Plugins" report.
[INFO] Generating "Continuous Integration" report.
[INFO] Generating "Source Repository" report.
[INFO] Generating "Project License" report.
[INFO] Generating "Mailing Lists" report.
[INFO] Generating "Plugin Management" report.
[INFO] Generating "Project Summary" report.
[INFO] [antrun:run {execution: id.site}]
[INFO] Executing tasks
     [echo] site phase
[INFO] Executed tasks
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESSFUL
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 3 seconds
[INFO] Finished at: Sat Jul 07 15:25:10 IST 2012
[INFO] Final Memory: 24M/149M
[INFO] ------------------------------------------------------------------------
```